DISKEX

2103-1 CODOS Disk File Exchange Utility Program

Release 1.0

USER MANUAL

APRIL, 1982

COPYRIGHT NOTICE 1981
Micro Technology Unlimited

This product is copyrighted. This includes the verbal description, programs, and specifications. The customer may only make BACKUP copies of the software for his own use. The copyright notice must be added to and remain intact on all such backup copies. This product may not be reproduced for use with systems which are sold or rented.

Copies may not be made for multiple internal use. In the event of the need for multiple copies to be used by the customer in his (her) own company or organization, volume discounts are available. In the case of large anticipated volume, licenses and royalties may be negotiated for the reproduction of the package.

Micro Technology Unlimited
2806 Hillsborough Street
P.O. Box 12106
Raleigh, NC 27605 USA
(919) 833-1458

NOTICE

Micro Technology Unlimited reserves the right to make changes to the product and specifications described in this manual at any time without notice.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

No warranties, either express or implied, are made by Micro Technology Unlimited with respect to this manual or the software described herein, its quality, performance, merchantability, or fitness for any particular application. This product is sold "as is". The buyer assumes all risk as to quality and performance. Under no circumstances will MTU be liable for direct, indirect, incidental or consequential damages resulting from any defect in the software, even if MTU has been advised of the possibility of such damages. Should the software prove defective following purchase, the buyer assumes the entire cost of all necessary servicing, repair or correction and any incidental, indirect, or consequential damages. You may have additional rights which vary from state to state, so that some of the above exclusions and limitations may not apply to you.

TRADEMARKS

A numer of trademarks are used throughout this manual. They include:

CP/M is a trademark of Digital Research
FLEX is a trademark of Technical Systems Cosultants (TSC)
IBM is a trademark of International Business Machines

TABLE OF CONTENTS

1. INTRODUCTION - 1

2. HARDWARE/SOFTWARE REQUIREMENTS - 2

3. PREPARATION OF STRANGER DISKS - 2

4. MAKING A WORK COPY OF DISKEX - 3

5. COPYING A STRANGER FILE TO CODOS - 4

6. DEVICES AVAILABLE IN DISKEX - 6

7. DISKEX COMMANDS - 7
   - 7.1 CODOS - 9
   - 7.2 COPY - 10
   - 7.3 DEF - 11
   - 7.4 DEN - 12
   - 7.5 DIR - 13
   - 7.6 DRIVE - 14
   - 7.7 DUMP - 14
   - 7.8 GO - 15
   - 7.9 FSIZE - 15
   - 7.10 NEXT - 16
   - 7.11 PRINT - 16
   - 7.12 READ - 17
   - 7.13 RESET - 17
   - 7.14 RESULT - 18
   - 7.15 SECTOR - 19
   - 7.16 SIZE - 20
   - 7.17 STATUS - 21
   - 7.18 TYPE - 22
   - 7.19 UNDEF - 22
   - 7.20 WRITE - 23

8. APPENDICES
   - A. ERROR MESSAGES - 24
   - B. CUSTOMIZATION - 25
   - C. ADDITIONAL NOTES ON THE DIFFERENT DISK FORMATS - - - - - - - - - - - - 26
   - D. PAGE ZERO USAGE - 27
   - E. HINTS AND SUGGESTIONS - 28

1. INTRODUCTION

The DISKEX program transfers text files from one type of disk to another. Its primary purpose is to allow disk files created on different computers to be converted to the disk format used by CODOS. In addition, it can be used to convert CODOS files to a format readable on other disk operating systems, or can translate disk formats between any two of the formats supported. The kinds of disks presently supported are shown in Table 1.

It is important to note that DISKEX is designed to transfer text files only. It will not transfer object code files programs since certain operations, such as tab expansion, are performed automatically during a transfer. This is no great loss since object code programs are rarely transportable to other system. The text files you transfer may represent programs or some other form of data. In the case of programs, modification of the program may be required before it will run successfully on the new system. Transferring the program with DISKEX simply saves retyping the bulk of the program.

To communicate with DISKEX, you enter commands from the Console, much the same as in CODOS. These commands allow you to declare the type of disks involved in the exchange, examine the disk directory, type files, etc.

To write a file on a CP/M, IBM, or FLEX disk, that file must already exist and be of sufficient size to hold the data you wish to write. The file does not have to exist if you are writing to a CODOS disk. Requiring that the file already exist is a slight inconvenience but makes the programming much simpler and more reliable. This is due to the fact that the different disk systems have a wide variety of schemes to allocate disk space to new files.

When DISKEX reads a file from a CP/M, IBM, and FLEX disks, a certain amount of translation will be done. For CP/M, tab characters will be expanded to the proper number of spaces. For IBM, EBCDIC will be translated to ASCII, and vice versa. Also, padding spaces will be removed when reading an IBM file, and added when writing an IBM file. For FLEX, space compression characters are expanded to the proper number of spaces.

When using DISKEX on an MTU-130 with one drive, the process of copying files requires the file be read into memory, and then written from memory to the destination file or device. The area of memory used is called the internal file buffer. This buffer determines the size of the files which may be copied on a single-drive system. There are more details on this in the examples given in section 5. The starting and ending addresses of this internal file buffer can be displayed using the STATUS command.

Table 1: Disk Formats Supported by DISKEX

System Format	DISKEX Name	Remarks
CODOS	CDS	MTU's operating system for 6502 computers.
CP/M	CPM	Digital Research's system for S-100 (8080 & Z-80) computers.
IBM	IBM	IBM Basic Data Exchange format 1. (128 byte sectors)
FLEX	FLX	TSC's system for 6800/6809 computers.

2. HARDWARE/SOFTWARE REQUIREMENTS

The hardware which comes standard in the MTU-130 satisfies the requirements for the DISKEX program. However, DISKEX makes use of the bottom three pages of memory in the floppy controller addresses C000 to C2FF hex for buffers. This area of memory normally holds the GRAPHDRIVER.Z software, which will be overwritten by DISKEX. If you exit DISKEX via a "CODOS" command, DISKEX will attempt to reload the GRAPHDRIVER.Z file. If this file is not on the disk in drive 0, or some other exit from DISKEX is taken, then you should manually reload the GRAPHDRIVER.Z file.

3. PREPARATION OF STRANGER DISKS

Any non-CODOS disk format is referred to as a "stranger" disk. If you are planning to copy a file from a stranger disk to a CODOS disk, no special preparation is necessary, unless you have only one disk drive. If you have only one disk drive, you will want to ensure that the file to be copied is short enough to fit in the DISKEX internal file buffer in its entirety. Longer files should be broken up into several short files on the stranger system.

If you are planning on transferring a file to a stranger disk, some advanced preparation will be necessary. The destination disk should be prepared on the stranger system by creating a "dummy" file which is long enough to hold the actual file you wish to transfer. This file will be overwritten by DISKEX during the transfer. The "dummy" file does not have to have the same name, nor does it have to be the same length. It must be at least as long as the actual file, however. Otherwise, only as much data will be written as will fit in the space allocated on your dummy file, and the copy will terminate with a "File I/O Error". You do not have to prepare a "dummy" file when copying to a CODOS disk.

For some additional notes concerning the various formats, refer to Appendix C.

4. MAKING A WORK COPY OF DISKEX

Before doing anything else, you should copy the DISKEX program from the distribution disk to a work disk. If you have a multiple-drive system, bring up a CODOS working disk in drive 0. Put the DISKEX Distribution disk in drive 1 and OPEN it. Now copy the DISKEX program to the disk in drive 0 by entering:

```
COPYF  DISKEX:1
```

CLOSE drive 1 and save the DISKEX distribution disk in its envelope and store it away in a safe place.

If you have a one-drive system, bring up a CODOS disk containing COPYF1DRIVE in the disk drive. Now start the copy process by entering:

```
COPYF1DRIVE
```

When you are prompted with:

```
PUT SOURCE DISK IN.
FILE (OR CR IF DONE)?=
```

place the DISKEX distribution disk into the drive and enter DISKEX as the file to copy. When you are prompted with:

```
PUT DEST. DISK IN.
CR WHEN READY.?=
```

place your work disk into the drive and enter a carriage return. Depending on the size of the large transient buffer, the copy may be complete at this point. If so the prompt will be:

```
PUT SOURCE DISK IN.
FILE (OR CR IF DONE)?=
```

Place the disk you wish to continue working with into the drive and enter a carriage return. Now you should save the DISKEX distribution disk in its envelope and store it away in a safe place.

If the copy has not completed, you will see the prompt:

```
PUT SOURCE DISK IN.
CR WHEN READY.?=
```

Note the difference in the second line of the prompt. You should place the DISKEX distribution disk into the drive again and enter a carriage return. When you are prompted with:

```
PUT DEST. DISK IN.
CR WHEN READY.?=
```

place your work disk into the drive and enter a carriage return. You will have to keep repeating the last two steps until you see the prompt:

```
PUT SOURCE DISK IN.
FILE (OR CR IF DONE)?=
```

At this point, place the disk you wish to continue working with into the drive and enter a carriage return. Now you should save the DISKEX distribution disk in its envelope and store it away in a safe place.

5. COPYING A STRANGER FILE TO CODOS

Once you have copied DISKEX onto your working disk, you may execute it by typing:

```
DISKEX
```

You should see a display similar to the following on your Console:

```
MTU DISKEX 1.0

Disk Exchange Program
Copyright 1981, Micro Technology Unlimited

DRIVE #0 28     R   2  ---
DRIVE #1 31   W   0    ---
DRIVE #2 02            ---
DRIVE #3 03            ---

Default Input  Drive = 1
Default Output Drive = 0

BEGINNING of internal file buffer $2D00
END of internal file buffer       $BDFF

DISKEX⟩
```

The meaning of this output is explained in the description of the STATUS command (Section 7.17). The "DISKEX⟩" displayed on the last line is the command prompt, which indicates DISKEX is ready to accept a typed command.

Here is an example of the steps which might be used to copy a text file from a CPM disk to a CODOS disk on a multiple drive system. First insert the stranger disk into drive 1. Next, you must define what kind of disk DISKEX should expect in each drive, and which drive is the source drive and which drive is the destination drive for the copy. In our example the commands would be:

```
DEF 0 CDS
DEF 1 CPM
```

To define drive 0 as a CODOS disk and drive 1 as a CP/M disk enter:

```
DRIVE 0 O
DRIVE 1 I
```

which defines drive 0 to be the destination (output) for the copy and drive 1 to be the source (input).

Next, check the directory of the stranger disk to locate the desired file:

```
DIR 1
```

lists the directory for drive 1. The actual format of the information displayed depends on the disk format selected. The display from a CPM disk might be:

```
 CP/M        E SZ     BAT ENTRYS

TEST    .ASM 0 06  02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
WUMPUS  .TXT 0 16  03 04 05 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Now you are ready to copy a file onto the CODOS disk. To copy the WUMPUS.TXT file to the CODOS disk, enter:

```
COPY WUMPUS.TXT  WUMPUS.T
```

This will copy the CP/M file "WUMPUS.TXT" from drive 1 to the CODOS file "WUMPUS.T" on drive 0. You must choose an existing file for the destination file for CP/M, FLEX, and IBM disks. On CODOS disks, the destination file may be a new file.

For 1 drive systems, the procedure is different. You will first have to read the source file into the internal file buffer. Then you switch disks and write the internal file buffer contents to the destination file. The following is an example of copying a CODOS text file to an IBM disk.

First, place the CODOS disk into the one drive. You may examine the file names on this disk by entering:

```
CODOS OPEN
```

and

```
CODOS FILES
```

To read the disk file MYDATA.D into the internal file buffer, enter the commands:

```
DEF 0 CDS
DRIVE 0 I
```

and

```
READ MYDATA.D
```

Next, place the IBM diskette into the drive and execute the commands:

```
DEF 0 IBM
```

and

```
DIR 0
```

to examine the files which are present on the disk. The following is an example of what you might see:

```
IBM

VOL1VNAME                                                          W
HDR1 DATA              00080 01001 73026      01811204            811204  01001
```

To copy the text in the internal file buffer to the file named DATA, enter the commands:

```
DRIVE 0 O
```

and

```
WRITE DATA
```

This completes the copy. This process implies that on single-drive systems, only files that will fit into the internal file buffer may be copied. Larger files will have to be broken into smaller pieces.

6. DEVICES AVAILABLE IN DISKEX

DISKEX supports its own device handling and drivers. Table 2 shows the list of devices available in DISKEX. The N (null), C (console), and P (printer) devices operate the same as in CODOS. (Note: you must patch a certain JMP instruction so DISKEX will access you printer driver routine. Refer to Appendix B for details.)

There are two devices which are not standard in CODOS. These are the M (memory) device, and the U (user) device. The M device reads and writes the internal file buffer. Characters will be read or written starting at the beginning of the buffer, and continuing until the end-of-file or the end of the buffer is reached. DISKEX will place a CNTL-Z (1A hex) at the end of the text data to indicate the end-of-file. This implies that $1A may not appear in the text files that pass through this buffer.

The U device is a user-defined device. This allows you to define you own custom device driver. You must provide the routines to input and output a character. You must then patch two JMP instructions so that DISKEX will use these routines to perform I/O for the U device. For details on how to make these patches, refer to Appendix B.

Table 2: Device Names

Name	Description
N	Null device
C	Console.
P	Printer.
M	Memory buffer.
U	User-defined device.

---

7. DISKEX COMMANDS

Shown in Table 3 is a list of the commands available in DISKEX. Obviously there are more commands than are needed to just copy text files. The commands typically involved in copying files are COPY, DEF, DIR, READ, UNDEF, and WRITE. These were used in the examples in section 5, except for UNDEF which is used to undefine a drive. Trying to read a file from a disk in an undefined drive will result in an error.

There are many additional commands which allow DISKEX to perform other functions. There are commands for displaying the contents and characteristics of files. These commands are PRINT, SIZE, and TYPE. There are commands to read and write individual sectors, which are DEN, SECTOR, and NEXT. Commands which involve the disk controller chip include RESET and RESULT.

Finally, there is the DUMP command to examine memory contents, and STATUS to display the status of DISKEX (the current type of disk in each drive, etc.). Detailed descriptions for each of the commands are contained the remainder of this section.

Table 3: DISKEX Commands

Command	Description
BOOT	Re-boot CODOS.
CODOS	Exit DISKEX to CODOS.
COPY	Copy a file from one disk to another.
DEF	Define the type of disk in a drive.
DEN	Set density parameter for disk.
DIR	List the file directory for a drive.
DUMP	Display contents of a memory block in hexadecimal.
GO	Execute a machine language program at a specified address.
PRINT	Print the contents of a file on a printer.
READ	Read a file from disk into memory.
RESET	Reset the floppy disk controller chip.
RESULT	Display floppy disk controller chip results.
SECTOR	Read a sector.
SIZE	Display the size of a file in memory.
STATUS	Display the status of the DISKEX program.
TYPE	Display the contents of a file on the Console.
UNDEF	"Undefine" a disk drive.
WRITE	Write a disk file from memory.

Along with the descriptions, a syntax diagram is given with each command. These syntax diagrams will use the same notation as used in the CODOS manual. The description of this notation is repeated here for convenience:

1. Angle brackets, "⟨" and "⟩", are used to enclose words describing the kind of entry required.

2. Square brackets, "[" and "]", are used to enclose optional arguments or symbols, which may be included or omitted as desired.

3. Ellipsis, "...", are used to indicate an arbitrary number of repetitions of the previous argument(s).

4. Symbols not enclosed in angle brackets are literal symbols which must be typed exactly as shown.

5. Braces, "{" and "}", are used to enclose each of several mutually exclusive choices, only one of which may be selected.

7.1 THE CODOS COMMAND

PURPOSE: To return to the CODOS Monitor, or pass a command to the CODOS Monitor for execution.

SYNTAX: CODOS [⟨command⟩]

ARGUMENTS:

⟨command⟩ - a CODOS Monitor command

DISCUSSION:

The CODOS command provides two functions. If a command is specified, then this command is passed to the CODOS Monitor for execution. Once the command has executed, execution returns to DISKEX. If no command is specified, then DISKEX exits to the CODOS Monitor.

EXAMPLES:

```
CODOS FILES 1
```

will execute a FILES command on drive 1.

```
CODOS DIR *.A
```

will execute the DIR utility, calling for all files with an extension of ".A" on the default drive to be listed.

NOTES:

1. If a CODOS Monitor command which executes from an overlay is specified, then the overlay must already be in memory or a CODOS disk with the overlay must be in drive 0.

2. If the command calls for a Utility or User Command to be executed, the program for that command must not interfere with the DISKEX machine code or page zero locations.

3. The command "CODOS" is the only proper way to exit DISKEX. Before returning to CODOS, DISKEX will attempt to OPEN drive 0 and load the GRAPHDRIVER.Z file from drive 0. If it is unable to OPEN drive 0, it will error back to CODOS. If it is unable to load the GRAPHDRIVER.Z file, it will print the message:

```
WARNING! GRAPHDRIVER.Z code has been overwritten by DISKEX's disk buffers.
Suggest you enter "GET GRAPHDRIVER.Z" to reload the graphics code.
```

Naturally, you will have to find a disk which has the GRAPHDRIVER.Z file on it in order to load the file.

7.2 THE COPY COMMAND

PURPOSE: To copy a file from a specified source to a specified destination.

SYNTAX: COPY [⟨filename⟩ [: ⟨drive⟩] | ⟨device⟩] [⟨filename⟩ : [⟨drive⟩] | ⟨device⟩]

ARGUMENTS:

⟨filename⟩ - a file name appropriate to the type of disk on which the file is found.
⟨drive⟩ - the desired drive. Defaults to the default input drive for the first file name. Defaults to the default output drive for the second file name.
⟨device⟩ - a device name.

DISCUSSION:

The COPY command will copy the file specified by the first argument to the file specified in the second argument. If a drive is not specified with a file name, then ⟨drive⟩ for the first argument defaults to the Input Drive, and the drive for the second argument defaults to the Output Drive. The Input and Output drives are specified by a DRIVE command.

EXAMPLES:

```
COPY  SOURCE.TXT:1  NEWSOURCE.T:0
```

will copy the text in the file "SOURCE.TXT" on drive 1 to the file "NEWSOURCE.T" on drive 0.

```
COPY  MYTEXT.T  C
```

will copy the text in the file MYTEXT.T on the Input Drive to the console device.

NOTES:

1. Before the copy will take place, a DEFINE command must have been issued for each drive involved in the copy.

7.3 THE DEF COMMAND

PURPOSE: To declare the type of disk in a specified drive.

SYNTAX: DEF ⟨drive⟩ ⟨type⟩

ARGUMENTS:

⟨drive⟩ - the desired drive.
⟨type⟩ - the type of disk to be placed in the indicated drive. See table below for disk types supported.

DISKEX Name	System Name	Remarks
CDS	CODOS	MTU's operating system for 6502 computers.
CPM	CP/M	Digital Research's system for S-100 (8080 & Z-80) computers.
IBM	IBM	IBM Basic Data Exchange format 1. (128 byte sectors)
FLX	FLEX	TSC's system for 6800/6809 computers.

DISCUSSION:

The DEF command is used to declare the type disk found in a particular drive. Commands which access data on a disk obtain the needed information on how the data on the disk is formated based on this declared type. The declaration for the specified drive will remain in effect until another DEF or UNDEF (UNDEFine) command is issued for the drive.

EXAMPLES:

```
DEF 1 IBM
```

will tell the system that the disk in drive 1 is an IBM disk.

```
DEF 0 CDS
```

will tell the system that the disk in drive 0 is a CODOS disk.

7.4 THE DEN COMMAND

PURPOSE: To set disk parameters to be used by the SECTOR and NEXT commands.

SYNTAX: DEN ⟨sides⟩ ⟨bytes/sector⟩ ⟨first⟩ ⟨last⟩ [⟨side⟩]

ARGUMENTS:

⟨density⟩ - the density, 1 = single density, 2 = double density.
⟨bytes/sector⟩ - a number specifying the bytes/sector (0 - 3). Use following table:

0 = 128 bytes/sect
1 = 256
2 = 512
3 = 1024

⟨first⟩ - first sector number on a track.
⟨last⟩ - last sector on a track.
⟨side⟩ - the side of the disk, 0 or 1. Defaults to side 0 if not specified.

DISCUSSION:

The DEN command is used to set disk parameters to be used by the SECTOR and NEXT commands. These parameters specify that a disk to be read is single or double density, the number of bytes per sector, the sector number of the first and last sectors on a track, and which side you wish to access.

EXAMPLES:

```
DEN 1 0 1 1A
```

specifies single density, 128 bytes/sector, the first sector per track is 1, the last sector per track is 1A hex (26 decimal), and the side to be accessed is side 0.

```
DEN 2 1 1 F 1
```

specifies double density, 256 bytes/sector, the first sector per track is 1, the last sector per track is 0F hex (15 decimal), and the side to be accessed is side 1.

NOTES:

1. When starting up DISKEX, the parameters set by the DEN command are initialized to those shown in the first example.

7.5 THE DIR COMMAND

PURPOSE: To display the directory of the disk in a specified drive.

SYNTAX: DIR ⟨drive⟩

ARGUMENTS:

⟨drive⟩ - the desired drive. Defaults to drive 0.

DISCUSSION:

The DIR command lists out all of the active files on the drive specified. Different disk types list out different information according to the standard for that disk type. If a drive is not given, drive zero will ALWAYS be assumed. The type of the disk in the drive should be the same as that given in the most recent DEF command for that drive.

EXAMPLES:

```
DIR
```

will list the files on the disk in drive 0.

```
DIR 1
```

will list the files on drive 1.

Here are examples of what is displayed for directories on CP/M, IBM and FLEX disks. For CP/M, the display might look like:

```
CP/M        E SZ     BAT ENTRYS

TEST    .ASM 0 06  02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
WUMPUS  .TXT 0 16  03 04 05 00 00 00 00 00 00 00 00 00 00 00 00 00
```

The file names contained in the directory are TEST.ASM and WUMPUS.TXT. For IBM, the display might look like:

```
IBM

VOL1VNAME                                                          W
HDR1 DATA              00080 01001 73026      01811204            811204  01001
```

This directory contains the single file name, DATA. For FLEX, the display might look like:

```
 FLEX              ST  END  SIZE   DATE
MYDATA
BACKUP   .BIN   00 0101 0408 0035  10/13/81
BAUDRATE.BAS   00 0409 040E 0006  11/24/81
TESTDATA.TXT   00 040F 0509 000A  01/04/82
```

The file names contained in the directory are BACKUP.BIN, BAUDRATE.BAS, and TESTDATA.TXT.

7.6 THE DRIVE COMMAND

PURPOSE: To declare the default input or default output drive.

SYNTAX: DRIVE ⟨drive⟩ [I|O]

ARGUMENTS:

⟨drive⟩ - the desired drive.

DISCUSSION

Specifying "I" or "O" will cause the specified drive to become the default input or output drive, respectively. The drive set as input will be used whenever a file is to be read, unless overridden by specifying a drive number on the input file name. The drive set for output will be used whenever a file is to be written, unless overridden by specifying a drive number on the file name. The initial defaults are: Input drive = 0 Output drive = 1

EXAMPLES:

DRIVE 0 I

sets drive 0 as the default input drive.

DRIVE 1 O

sets drive 1 as the default output drive.

7.7 THE DUMP COMMAND

PURPOSE: To display the specified portion of memory.

SYNTAX: DUMP [⟨from address⟩ [⟨to address⟩]]

ARGUMENTS:

⟨from address⟩ - address of first byte to display. Defaults to the memory address of the disk buffer used in SECTOR command, which is C000 hex.

⟨to address⟩ - address of last byte to display. Defaults to ⟨from address⟩ plus 80 hex (8 lines worth).

DISCUSSION

The DUMP command prints hex and ASCII dumps of memory. The output of the DUMP command is formatted with each line containing the address followed by 16 hex bytes, followed by the ASCII representation for the 16 hex bytes. A period is printed in the ASCII section for byte values outside the range 20 to 7F hex.

EXAMPLES:

DUMP 3000 3100

will display memory contents from 3000 hex to 3100 hex.

DUMP

will display C000 to C080 hex.

7.8 THE GO COMMAND

PURPOSE: To execute a machine language subroutine from DISKEX.

SYNTAX: GO ⟨address⟩

ARGUMENTS:

⟨address⟩ - the address at which to begin execution of a machine langauge program.

The GO command is used to execute a machine language program by executing a JSR to the specified address. Since the machine language program will be executed as a subroutine, the program may return to the DISKEX program with an RTS instruction.

EXAMPLES:

GO B000

will execute the machine language subroutine which starts at address B000 hex.

7.9 THE FSIZE COMMAND

PURPOSE: To determine the number of bytes a file will occupy in CODOS format.

SYNTAX: FSIZE ⟨filename⟩ [: ⟨drive⟩]

ARGUMENTS:

⟨filename⟩ - a file name appropriate to the type of disk on which the file is found.
⟨drive⟩ - the desired drive. Defaults to the default input drive.

DISCUSSION:

The FSIZE command is used to determine the size of a file. This size reflects the number of bytes the file would occupy on a CODOS disk. The number printed is the number of characters (bytes in HEX) of the specified file. This number may be different from the actual number of bytes the file occupies on a CPM, IBM, or FLEX disk. In the case of CP/M and FLEX files, the difference is due to the expanding of tab characters. In the case of IBM files, trailing spaces are removed from the records, and are thus not included in the total.

EXAMPLES:

FSIZE DATA1:1

will display the size of the file named DATA1 on drive 1. The size displayed might be:

File Size = $00000966

FSIZE WP.TXT

will display the size of the file name WP.TXT on the default input drive.

7.10 THE NEXT COMMAND

PURPOSE: To read the next sequential sector into memory.

SYNTAX: NEXT

ARGUMENTS:

none

DISCUSSION:

The NEXT command executes a SECTOR read command for the next sequential sector. This saves retyping the SECTOR command when you wish to read the next sector. The sector will be read into memory at C000 hex. The NEXT command will also display the track and sector number of the sector read.

Typically a DEN and SECTOR read command are executed prior to execution of a NEXT command. If no SECTOR command has been previously executed, the NEXT command will read track 0, sector 1, as the next sector. If no DEN command has been previously executed, the DEN parameters will be at their default values of single density, 128 bytes sectors, and sectors numbered 1 to 26 hex on a track.

EXAMPLE:

NEXT

will read the next sequential sector from the disk last read.

7.11 THE PRINT COMMAND

PURPOSE: To print the specified file on the printer.

SYNTAX: PRINT ⟨filename⟩ [: ⟨drive⟩]

ARGUMENTS:

⟨filename⟩ - a file name appropriate to the type of disk on which the file is found.
⟨drive⟩ - the desired drive. Defaults to the default input drive.

DISCUSSION:

The PRINT command is used to copy a specified file to the printer. It is identical to a COPY command where the printer is specified as the output destination. This command assumes you have made the patch described in Appendix B to interface a printer driver routine to DISKEX.

EXAMPLES:

PRINT WUMPUS.TXT

will output the file WUMPUS.TXT on the default input drive to the printer.

7.12 THE READ COMMAND

PURPOSE: To input a file into the internal file buffer.

SYNTAX: READ $\left\{\begin{matrix}\langle\text{filename}\rangle \\ \langle\text{device}\rangle\end{matrix}\right.$ $\left[: \langle\text{drive}\rangle\right]$

ARGUMENTS:

⟨filename⟩ - a file name appropriate to the type of disk on which the file is found.
⟨drive⟩ - the desired drive. Defaults to the default input drive.
⟨device⟩ - the desired device.

DISCUSSION:

The READ command is used to read a file into the internal file buffer. This will abort should the file buffer become full before the end of the file is reached. The READ command is identical to COPY command where the M device is specified as the output destination.

EXAMPLES:

```
READ  DATA.TXT
```

will read the file DATA.TXT on the default input drive into the internal file buffer.

```
READ  U
```

will read a file from the user device (assuming one has been defined, see Appendix B) into the internal file buffer.

7.13 THE RESET COMMAND

PURPOSE: To reset the NEC-765 Floppy Controller chip.

SYNTAX: RESET

ARGUMENTS:

none

DISCUSSION:

The RESET command resets the NEC-765 Floppy Controller chip . This command should be executed after an "FDC Command" or "FDC Result" error occurred.

EXAMPLES:

```
RESET
```

will reset the floppy controller chip.

7.14 THE RESULT COMMAND

PURPOSE: To display the status registers of the NEC-765 Floppy Controller chip.

SYNTAX: RESULT

ARGUMENTS:

none

DISCUSSION:

The RESULT command displays the contents of the NEC-765 Floppy Controller's status registers. This will be the result status from the last command issued to it. This command can help determine the exact nature of errors that occur while trying to read a disk. The registers are printed out in the following form:

```
MSR=80 ST-0=40 ST-1=80 ST-2=00  C=0C R=01 H=00 N=01
```

where

```
MSR  =  Main Status Register
ST-X =  Status register X
C    =  Cylinder  (track)
R    =  Record    (sector)
H    =  Head      (side)
N    =  Bytes/sector (see table in DEN command)
```

See the data sheets on the NEC-765 in the Disk Controller hardware manual for more information on these registers.

EXAMPLES:

```
RESULT
```

will display the contents of the floppy chip's status registers.

7.15 THE SECTOR COMMAND

PURPOSE: To read or write a specifed sector.

SYNTAX: SECTOR [R / W] ⟨drive⟩ ⟨track⟩ ⟨sector⟩

ARGUMENTS:

R = when specified as the first argument calls for the sector to be read.
W = when specified as the first argument calls for the sector to be written.

⟨drive⟩ = the desired drive.
⟨track⟩ = the desired track.
⟨sector⟩ = the desired sector.

DISCUSSION:

The SECTOR command is used to read a specified sector into memory. This command allows you to investigate disks of other formats not supported by DISKEX. It also allows you try to read and possibly correct bad sectors on a disk. Care should be exercised when attempting this. A DEN command should be executed prior to first use of the SECTOR command to specify the characteristics of the disk to be read.

The location of the sector buffer used by the command is C000 hex. (This buffer is also used when reading and updating the directories while accessing disk files.) For convenience, C000 hex is also the default address for the DUMP command.

EXAMPLES:

```
SECTOR W 0 10 6
```

will write sector 6 of track 10 (16 decimal) on the disk in drive 0.

```
SECTOR R 1 2 3
```

will read sector 3 of track 2 of the disk in drive 1. To display the first 80 bytes of the data read, enter the command:

```
DUMP
```

7.16 THE SIZE COMMAND

PURPOSE: To display the size of the file contained in the internal file buffer.

SYNTAX: SIZE

ARGUMENTS:

none

DISCUSSION:

The SIZE command displays the size of the file contained in the internal file buffer, along with addresses of where this file resides in memory. The size is displayed as follows:

SIZE $xxxx (dd K) @ $xxxx - $xxxx $xxxx

The first number is the number of bytes in the file, given in hex. The number in parentheses is the number of K (1024 byte blocks) in the file, rounded up to the nearest K, given in decimal. The next two addresses are the start and end address of the file currently in memory, and last address is the end of the internal file buffer.

The size is determined by searching for an end-of-file character (which is a CNTL/Z, or 1A hex). If the memory within the internal file buffer has been used for other purposes, there may not be a valid end-of-file character in the buffer. In this case, the size displayed by the SIZE command will be erroneous.

EXAMPLES:

SIZE

will display the size and address for the file currently in the internal file buffer.

7.17 THE STATUS COMMAND

PURPOSE: To display the status of all drives and certain DISKEX parameters.

SYNTAX: STATUS

ARGUMENTS: none

DISCUSSION:

The STATUS command will display the status of drives 0 through 4, along with certain DISKEX parameters. A typical display might be as follows:

```
DRIVE #0 28     R   2  CDS
DRIVE #1 31   W   0    IBM
DRIVE #2 02            ---
DRIVE #3 03            ---

Default Input  Drive = 1
Default Output Drive = 0

BEGINNING of internal file buffer $2D00
END of internal file buffer       $BDFF
```

The hex number following the drive number is the status byte returned by the Floppy Controller chip for that drive. The information in this byte is displayed in the next four columns. If the letter shown below is present in its column, the condition is true. If a blank is present in that column, then the condition is false.

Column	Letter	Meaning
1	W	the disk is write protected
2	R	the drive is READY
3	0	the drive head is currently at track 0
4	2	the disk is a two sided disk

The display above shows that drive 0 is ready, and contains a two sided disk which is not write protected. Drive 1 is not ready (i.e. contains no disk, or the drive door isn't closed), and the disk head is at track 0. The write protect indication is a result of no disk being present in the drive, but does serve to indicate that drive 1 is responding and is therefore present in the system. There is no indication for drives 2 and 3, not even write protect. This indicates that these drives are not present in the system.

To the right of the drive status information is a column showing what type of disk has been defined for the drive. In the display above, drive 0 contains a CODOS disk, and drive 1 contains an IBM DISKEX. The "---" for drive 2 and 3 indicates the disk type for these drives has not be defined. The DEF command is used to set or change the type for any of the drives. The UNDEF command is used to set the type as undefined.

Also displayed is the default input and default output drives, along with the beginning and end of the internal file buffer.

EXAMPLES:

```
STATUS
```

will display the status of each of the drives and the DISKEX parameters.

7.18 THE TYPE COMMAND

PURPOSE: To type a file on the console device.

SYNTAX: TYPE ⟨filename⟩ [: ⟨drive⟩]

ARGUMENTS:

⟨filename⟩ - a file name appropriate to the type of disk on which the file is found.
⟨drive⟩ - the desired drive. Defaults to the input drive.

DISCUSSION:

The TYPE command types the specified file on the console device. This is identical to executing a COPY command with the C device specified as the destination.

EXAMPLES:

```
TYPE  DATA1:0
```

will type the file named DATA1 on drive 0 to the console.

7.19 THE UNDEF COMMAND

PURPOSE: To set the disk type for a drive as "undefined".

SYNTAX: UNDEF ⟨ drive ⟩

ARGUMENTS:

⟨drive⟩ - the desired drive.

DISCUSSION:

The UNDEF command is used set the disk type for a drive as "undefined". Trying to access files on a drive which is undefined will result in a "Drive not Defined" error. The STATUS command displays "---" in the rightmost column to indicate that the drive type is undefined.

When first starting up DISKEX, all drives are initialized to the undefined state. This requires that you execute a DEF command before you can access files on a disk in that drive. When you remove a disk from a drive, it would be safest to execute an UNDEF command to undefine the drive. This will help prevent accidentally writing the wrong format on a disk placed in this drive later.

EXAMPLES:

```
UNDEF 0
```

will set the disk type for drive 0 as undefined.

7.20 THE WRITE COMMAND

PURPOSE: To write the file contained in the internal file buffer to disk.

SYNTAX: WRITE $\left\{\begin{matrix}\langle\text{filename}\rangle \\ \langle\text{device}\rangle\end{matrix}\right\}$ $\left[: \langle\text{drive}\rangle\right]$

ARGUMENTS:

⟨filename⟩ - a file name appropriate to the type of disk on which the file is found.
⟨drive⟩ - the desired drive. Defaults to the output drive.
⟨device⟩ - a device name.

DISCUSSION:

The WRITE command writes the file currently in the internal file buffer into the specified disk file. This is identical to a COPY command where the M device is specified as the source file.

EXAMPLES:

```
WRITE  NEWFILE.T
```

will write the file contained in the internal file buffer to the file named NEWFILE.T.

```
WRITE  C
```

will write the file contained in the internal file buffer to the console device.

APPENDIX A

ERROR MESSAGES

#	Name	Meaning
$01	Command Error	Undefined command.
$02	Invalid Disk Type	Disk type does not exist.
$03	Invalid Drive #	Drive number is not 0, 1, 2, or 3.
$04	Memory Full	Memory Full on file input.
$05	Seek Error	Drive malfunction on seek.
$06	Recalibrate Error	Drive malfunction on recalibrate.
$07	(Not used)	-
$08	User Device Does Not Exist	USR device has not been installed.
$09	(Not used)	-
$0A	Invalid Density Error	Density other than 1 or 2.
$0B	Not Implemented Yet	Feature is not implemented yet.
$0C	File I/O Error	I/O error. probably on write.
$0D	Read Sector Error	Error on reading from disk.
$0E	Write Sector Error	Error on writing to disk.
$0F	FDC Command Error	Error during command phase of disk command.
$10	FDC Result Error	Error during result phase of disk command.
$11	Open Error, File Not Found	Could not find specified file
$12	Close Error, File Not Closed	Error while trying to close an output file.
$13	Invalid File name	Name not valid in current disk type.
$14	Invalid Char in Number	Non hex char in input string.
$15	Missing Parameter	A needed value is missing.
$16	DMA Page Error	DMA page other than Disk Controller memory.
$17	Invalid device	Undefined device name.
$18	Define Error	Error on define command, disk possibly wrong type.
$19	Drive not defined	Disk drive has not been defined.
$1A	(Not used)	-
$1B	Boot error	Disk in drive 0 is not a CODOS disk.
$1C	Invalid number	Number too large for parameter.
$1D	Invalid parameter	Parameter is not of possible choices.
$1E	Memory buffer error	Error in the specified limits of the memory buffer. End probably less than beginning.
$FF	(Internal) Invalid Error #	The system probably crashed. RE-LOAD (memory error or system crash) The error routine was called with an invalid error number.

APPENDIX B

ADDING A PRINTER AND USER DEVICE

To make DISKEX support a printer or a user device, you must make some small patches to the DISKEX program. If you want the patches permanent, you may use the following procedure. First enter the command:

```
GET DISKEX
```

to load the DISKEX program into memory. Next you can use the CODOS SET and DUMP commands to install and check your patches. Then you should enter the command:

```
GETLOC DISKEX
```

to display the load addresses of the DISKEX program. You should then use the CODOS save command to write the DISKEX program in it entirety, plus any of your added code to a new file. You might write it to a file named NEWDISKEX.C. Once you have saved the new version you should execute it to test out your patches.

To add a printer driver to DISKEX you must patch the JMP instruction at location 812 hex to jump to a printer output routine. This routine should output the character in the 6502 Accumulator to the printer. The output routine generated by the SYSGENPRINTR routine meets this requirement. To use this routine, enter the following CODOS command to make the needed patch:

```
SET 812 4C 80 D2
```

To add a user device, you must supply an input driver if you want to read from this device, and an output driver if you want to write to this device. The first consideration would be where to put the extra code for the routines. The page of memory just preceeding the DISKEX program (700 to 7FF hex) is available. Another alternative is move the beginning of the internal file buffer up to make room for the routines. This is done by patching the default buffer pointer at location 821 hex. For example, to patch the beginning of the internal file buffer to 3000 hex, enter the command:

```
SET 821 00 30
```

You should put your code starting at the original beginning of the internal buffer. There is some space between this location and end of the DISKEX program. This is used for variable storage, and must not be disturbed.

For the user device input driver, you must supply a routine which will input a character to the 6502 Accumulator and return the 6502 Carry flag clear if this is successful. If the end-of-file is reached or an input error occurs, the routine should return the Carry flag set. Assuming you put this routine at 700 hex, you would enter the following CODOS command to make the patch:

```
SET 815 4C 00 07
```

For the user device output driver, you must supply a routine to output the character in the 6502 Accumulator. If the operation is successful, the routine should return the 6502 Carry flag clear. If an output error occurs, it should return the Carry flag set. To link this routine to DISKEX, patch the JMP instruction at location 818 hex. Assuming you put this routine at 780 hex, you would enter the following CODOS command to make the patch:

```
SET 818 4C 80 07
```

APPENDIX C

ADDITIONAL NOTES ON THE DIFFERENT DISK FORMATS

The following is some additional information dealing with specific details on the various disk formats supported by DISKEX.

CODOS

For CODOS, the system SVCs are used except in the DIRECTORY command. DISKEX does its own directory listing of the CODOS files.

NOTE: SVCs are NOT used unless a CODOS disk is to be read or written or a CODOS command is sent to CODOS via the CODOS command.

CP/M

Tabs are expanded into spaces on read, but spaces are NEVER compressed back to tabs on write. Multiple Extents are supported. The sector staggering is that from CP/M 1.3 and later. (1.2 is usually the same as 1.3).

IBM

When writing to IBM, it is best to write to a new disk since a new disk has all of the space allocated to the first file. When an IBM file is closed after writing, the EOD (End of Data) is updated to reflect the size of the data set just written.

Also, the IBM format is very slow on reading and writing so do not be alarmed if it is taking more time than you thought it would. IBM is the SLOWEST of all of the disk formats.

FLEX

FLEX random access files are treated like sequential files, with the first sector (the sector map) being completely ignored. Output files DO NOT have to originally contain text. If you want, you can output to a .COM file. The advantage of a .COM file is that it is easier to create (using the save command) Be sure to allow enough room for the new file. Unlike FLEX, DISKEX can write to already existing files!!

APPENDIX D

PAGE ZERO USAGE

In addition to the page zero locations allocated to CODOS and SVC's, DISKEX uses the following additional page zero locations:

ADDRESS	PURPOSE
A0	General memory pointer.
A2	Temporary pointer used while printing.
A4-A7	Scratch storage.

APPENDIX E

HINTS AND SUGGESTIONS

For small files it is best to read them into the internal file buffer and then write them back out from the buffer. This may be done with the READ and WRITE commands.

It is also a good idea to use high quality disks. The reason for this is that DISKEX only does a few (3) retries on writing and only a few more (5) on reading.

Leave all disks write protected except when you are going to actually write to them.